ИНФОРМАТИКА

2009 **январь-март** **№ 1**

УДК 004.93'1; 004.932

А.И. Кравчонок

АЛГОРИТМЫ МЕДИАННОЙ ФИЛЬТРАЦИИ С ОКНОМ 3×3 НА ОСНОВЕ НЕПОЛНЫХ СОРТИРУЮЩИХ СЕТЕЙ

Предлагаются новые алгоритмы медианной фильтрации с окном 3×3, основанные на использовании неполных сортирующих сетей и позволяющие эффективно вести поиск медианы элементов окна фильтра. Описываются алгоритмы, которые позволяют значительно увеличить скорость выполнения медианной фильтрации для окна фильтра 3×3 на персональном компьютере и делают возможным ее применение для широкого круга задач обработки изображений в режиме реального времени. Проводится сравнительный анализ различных алгоритмов медианной фильтрации с точки зрения количества необходимых для их выполнения операций и быстродействия.

Введение

Медианная фильтрация хорошо зарекомендовала себя в различных задачах, связанных с обработкой изображений [1, 2]. Преимущество медианной фильтрации (например, по сравнению с линейными фильтрами) заключается в том, что она не вызывает значительного размытия изображения в области границ. Весьма перспективными являются адаптивные медианные фильтры, позволяющие варьировать размеры окна медианного фильтра в зависимости от различных параметров фильтруемого участка изображения, тем самым регулируя степень его размытия [2, 3]. Однако нелинейность медианного фильтра отрицательно сказывается на скорости его работы. Например, обработка последовательности изображений, поступающих с видеокамеры, при помощи медианного фильтра требует большого количества времени, и применять этот фильтр в режиме реального времени становится нецелесообразно. На данный момент известны алгоритмы [4, 5], основанные на модифицированном алгоритме Хуанга [6] и позволяющие выполнять медианную фильтрацию изображений для произвольных размеров окон фильтра практически без падения производительности при увеличении их размеров. Вместе с тем минимальное время выполнения таких алгоритмов (даже для небольших окон) не позволяет применять их в режиме реального времени для обработки видеопоследовательностей. При этом для задач обработки последовательностей изображений, поступающих с видеокамеры в режиме реального времени (например, задачи поиска движущихся объектов и их отслеживания), нет необходимости выполнять медианную фильтрацию с окнами большого размера, так как более крупные окна лишь удаляют более крупные детали и сильнее размывают изображение. Размер изображений, поступающих с видеокамеры, зачастую не превышает 720×576 пикселей, и наиболее подходящим окном медианного фильтра для обработки таких изображений часто является окно 3×3. Поэтому алгоритм, пусть и не универсальный (подходящий для произвольных размеров окон), но эффективно работающий на небольших окнах фильтра, будет полезен для применения в задачах подобного типа.

Многие алгоритмы медианной фильтрации используют различные типы сортировки элементов окна фильтра. Среди них можно выделить алгоритмы, основанные на обменных сортировках, преимуществом которых, как правило, является возможность их реализации без использования условных операторов [7, 8], хотя верно это не для всех сортировок подобного типа (например, сложно реализовать без условий быструю сортировку Хоара [9, 10]). Реализация без условий достигается путем замены операторов условного перехода арифметическими действиями, что значительно ускоряет выполнение алгоритма на современных процессорах [7, 11–15]. На данный момент практически все процессоры имеют конвейерную организацию и оптимизированы для выполнения линейных программ. Наличие в программе ветвлений (операторов условного перехода) делает невозможной полную загрузку конвейера процессора, так как неизвестно, какую часть команд программы необходимо выполнять в дальнейшем. Для решения данной проблемы и во избежание простоя конвейера в процессоре реализован механизм предсказания ветвлений, который, очевидно, не всегда может угадывать правильно ветвь програм-

мы, которая будет выполняться в дальнейшем. Таким образом, уменьшение, а в идеале полное удаление команд условного перехода из программы, влечет значительное повышение скорости ее выполнения [11–15]. Алгоритм сортировки последовательности чисел, основанный на сравнении пары чисел и возможном их обмене в случае нарушения некоторого порядка, может быть реализован посредством арифметических операций, что ускорит выполнение программы, реализующей этот алгоритм. Сортирующие сети как раз основаны на операции сравнения-обмена, а следовательно, легко реализуются на персональном компьютере без условных операторов. Поэтому сортирующие сети могут быть полезны не только для аппаратных и параллельных реализаций поиска медианы последовательности, но и для реализаций на персональных компьютерах. Преимуществом алгоритмов на основе сортирующих сетей также является то, что они могут работать с числами произвольной величины, в то время как алгоритм Хуанга и его улучшенные версии [4–6] допускают работу только с числами на отрезке [0...255] вследствие использования в алгоритме гистограммы для сортировки этих чисел [16]. Применение сортирующих сетей, в том числе и неполных, для реализации медианной фильтрации рассматривалось и ранее [8, 17], но в основном в контексте аппаратной реализации. Задача эффективной реализации неполных сортирующих сетей на персональном компьютере рассматривалась не столь полно и подробно. Кроме этого, на данный момент можно предложить более эффективные варианты неполных сортирующих сетей, описанные в данной статье и позволяющие быстрее находить медиану окна фильтра для двухмерных изображений.

Среди алгоритмов поиска медианы последовательности элементов 3×3 окна фильтра, основанных на операциях сравнения-обмена элементов, можно выделить следующие алгоритмы: Паеса (20 операций сравнения-обмена) [16], неполной сортировкой прямым выбором (16,5/15 операций сравнения-обмена для двойной/четверичной сортировки) [7], Кучеренко и Очина (приблизительно 13 операций сравнения-обмена) [17]. Данные алгоритмы, хотя и проигрывают по количеству операций сравнения другим, например алгоритму слияния упорядоченных столбцов (около 9 операций сравнения на медиану) [18, 19], но за счет реализации без использования условных операторов практически не уступают, а часто и превосходят их по скорости работы.

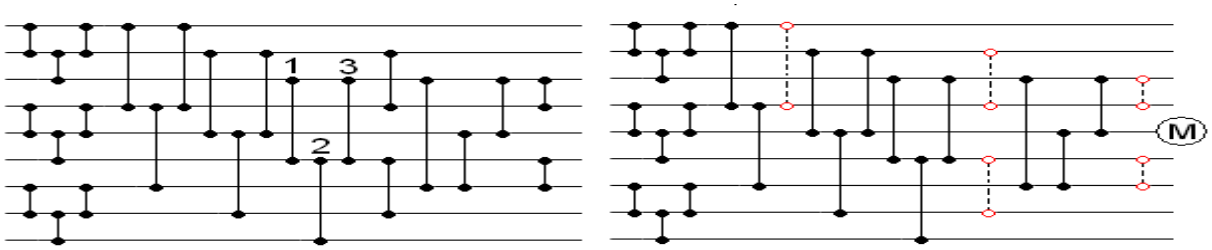
1. Сортирующие сети

Сортирующие сети относятся к сравнивающим сетям, в основе которых лежит операция сравнения-обмена. Будем считать, что подобную операцию реализует некое устройство – компаратор, или сравнивающее устройство. Компаратор имеет два входа и два выхода. На вход компаратора поступают два числа, на выходе получаем эти же два числа, но переупорядоченные в соответствии с неким порядком (по возрастанию либо по убыванию). Таким образом, компаратор – это устройство, которое имеет два входа, x и y, два выхода, x' и y', и выполняет операцию $x' = \min(x, y)$, $y' = \max(x, y)$, если задан порядок по возрастанию, или $x' = \max(x, y)$, $y' = \min(x, y)$, если задан порядок по убыванию [10, 20–23].

При помощи компараторов и соединяющих их проводов строятся *сравнивающие сети* [20–22]. *Сортирующая сеть* – это сравнивающая сеть, в которой выходная последовательность является монотонно неубывающей. В сортирующих сетях строго фиксируется последовательность сравнений, которая не зависит от особенностей входных данных, что допускает линейную программную реализацию сортирующих сетей без использования ветвлений. Для поиска медианы последовательности нет необходимости сортировать все элементы этой последовательности, поэтому сети, которые выполняют поиск медианы, не сортируя полностью входную последовательность элементов, будем называть неполными сортирующими сетями. Сортирующие сети принято изображать в виде набора горизонтальных линий (проводов), на вход которых поступают обрабатываемые элементы и продвигаются по ним вправо, пока не достигнут выхода. Компараторы изображаются в виде вертикальных линий, соединяющих соответствующие провода, и обозначают выполнение в данном месте сортирующей сети операции сравнения-обмена над элементами, продвигающимися по этим проводам.

1.1. Сеть поиска медианы на основе сортирующей сети Флойда

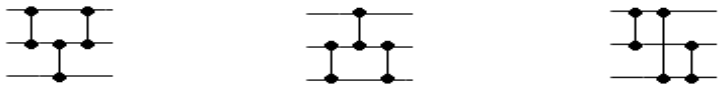
Возьмем за основу одну из сортирующих сетей для девяти элементов, предложенную Флойдом в 1964 г. (рис. 1, *а*) [10]. Сеть позволяет отсортировать 9 элементов за 25 операций сравнения-обмена, т. е. содержит 25 компараторов. Отбросив некоторые компараторы из этой сети, которые, очевидно, не влияют на положение медианы входной последовательности, получим неполную сортирующую сеть, изображенную на рис. 1, *б* (ненужные компараторы обозначены пунктирной линией), которая позволяет находить медиану за 20 операций сравнения-обмена, что аналогично количеству операций в алгоритме Паеса [16]. Корректность работы полученной сети легко проверить полным перебором всех возможных комбинаций входных данных и проверкой полученного результата, причем нуль-единичный принцип [10, 20, 21] позволяет значительно сократить количество проверяемых вариантов.



а) *б)*

Рис. 1. Сортирующие сети: *а)* сортирующая сеть Флойда;
б) неполная сортирующая сеть для поиска медианы

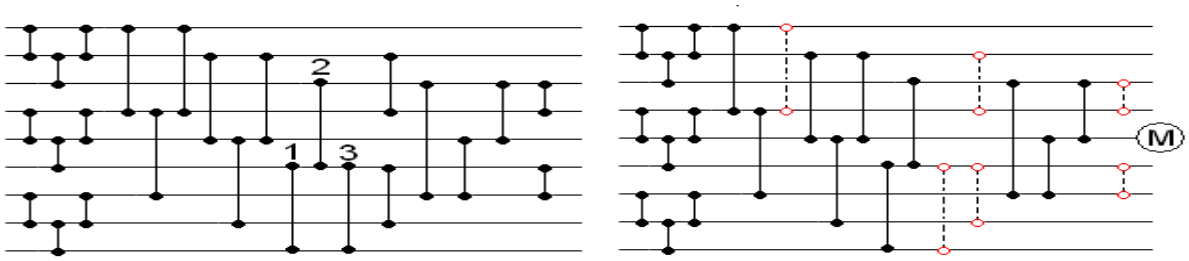
Изображенная на рис. 2, *а* сеть – это сортирующая сеть для трех элементов. Заметим, что возможны другие варианты сортирующих сетей для трех элементов (рис. 2, *б*, *в*).



а) *б)* *в)*

Рис. 2. Различные варианты сортирующих сетей для трех элементов

Заменим тройку компараторов, обозначенных номерами 1–3 на рис. 1, *а*, на другую тройку, также сортирующую три элемента. Получим сеть, изображенную на рис. 3, *а* (эквивалентную сети Флойда), с тем же количеством компараторов. Теперь становится очевидным, что можно удалить еще один компаратор, обозначенный на рис. 3, *а* номером 3, который не влияет на положение медианы.



а) *б)*

Рис. 3. Сортирующие сети: *а)* видоизмененная сортирующая сеть Флойда;
б) неполная сортирующая сеть для поиска медианы

Таким образом, получена сеть поиска медианы последовательности из 9 элементов, которая содержит 19 компараторов и при программной реализации потребует на одну операцию сравнения меньше, чем алгоритм Паеса. Заметим, что 19 операций сравнения-обмена являются необходимым минимумом для поиска медианы последовательности из 9 элементов при помощи сортирующей сети [3, 16].

Легко доказывается, что полученная сеть находит именно медиану. На первом шаге сортируем три тройки чисел; обозначим отсортированные тройки: $A=(a_1,a_2,a_3)$, $B=(b_1,b_2,b_3)$, $C=(c_1,c_2,c_3)$, где $a_1\le a_2\le a_3$, $b_1\le b_2\le b_3$, $c_1\le c_2\le c_3$. Проанализируем, какие из этих чисел могут быть медианой. Если расположить тройки чисел друг под другом, то получим 3×3 матрицу M:

$$M=\begin{pmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{pmatrix}.$$

Отсортируем тройки (a_1,b_1,c_1), (a_2,b_2,c_2), (a_3,b_3,c_3) и получим отсортированные тройки $(a_1^s,b_1^s,c_1^s),(a_2^s,b_2^s,c_2^s),(a_3^s,b_3^s,c_3^s)$. Заметим, что после такой сортировки выполняются равенства $a_1^s\le a_2^s\le a_3^s$, $b_1^s\le b_2^s\le b_3^s$, $c_1^s\le c_2^s\le c_3^s$. Следует это из того факта, что если $a<b$ и $c<d$, то $\min(a,c)<\min(b,d)$ и $\max(a,c)<\max(b,d)$. Указанные неравенства можно расширить на упорядоченные последовательности с большим количеством элементов.

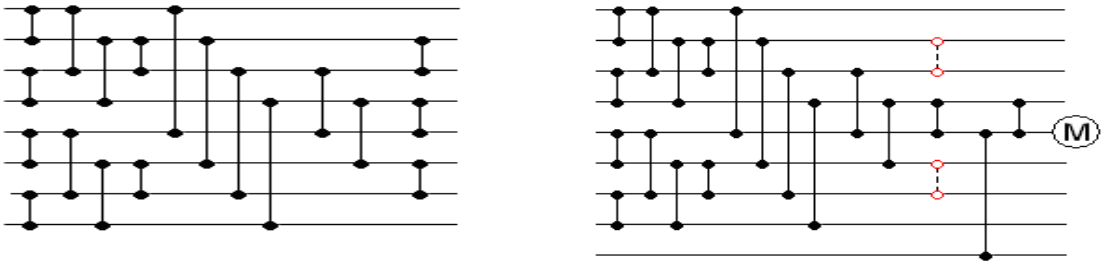
Тогда a_1^s не может быть медианой, так как $a_1^s<a_2^s$, $a_1^s<a_3^s$, $a_1^s<b_1^s$, $a_1^s<b_2^s$, $a_1^s<b_3^s$, а медиана девяти элементов не может быть меньше пяти элементов. Элемент b_1^s тоже не может быть медианой вследствие того, что $b_1^s<b_2^s$, $b_1^s<b_3^s$, $b_1^s<c_1^s$, $b_1^s<c_2^s$, $b_1^s<c_3^s$. Аналогично доказывается, что каждый из элементов a_2^s,b_3^s,c_2^s,c_3^s также не может быть медианой.

Из вышесказанного следует, что медианой из первой тройки чисел (a_1^s,b_1^s,c_1^s) может быть только максимум этой тройки, из третьей тройки (a_3^s,b_3^s,c_3^s) – только минимум тройки чисел, а из средней тройки (a_2^s,b_2^s,c_2^s) медианой может быть только средний элемент.

Минимум и максимум тройки чисел можно найти за две операции сравнения-обмена, а средний элемент – за три операции, что видно в предложенной сортирующей сети. В конце вычисленные три элемента – потенциальные медианы – сортируются (три операции сравнения-обмена), средний элемент отсортированной тройки (a_3^s,b_2^s,c_1^s) и есть искомая медиана.

1.2. Сеть поиска медианы на основе нечетно-четной сортирующей сети Бэтчера

Рассмотрим нечетно-четную сортировку Бэтчера для восьми элементов (рис. 4, *а*) [22].



а) *б)*

Рис. 4. Сеть поиска медианы на основе нечетно-четной сортирующей сети Бэтчера:
а) нечетно-четная сортирующая сеть Бэтчера для восьми элементов;
б) неполная сортирующая сеть поиска медианы на основе сети Бэтчера

Нечетно-четная сортировка Бэтчера основана на процедуре слияния отсортированных на предыдущем шаге частей последовательности. На рис. 4, *а* изображена сортирующая сеть для восьми элементов. Заметим, что вставку последнего девятого элемента в сеть Бэтчера для восьми элементов можно выполнить за две операции сравнения-обмена, так как необходимо лишь проверить, не является ли вставляемый элемент медианой, и если не является, то он больше либо меньше медианы. Для этого сравниваем новый элемент с пятым элементом отсортированной последовательности из восьми элементов. В случае если новый элемент больше пятого, то пятый элемент и есть медиана; если же новый элемент меньше, то последующее сравнение с четвертым элементом либо оставит новый элемент на месте, либо сдвинет его выше. В любом из этих случаев медиана девяти элементов будет находиться на пятом проводе сети.

После удаления компараторов, не влияющих на положение медианы, а также после добавления еще двух компараторов, вставляющих девятый элемент в сеть из восьми исходных, получим сеть, изображенную на рис. 4, *б*, которая, как и найденная ранее на основе сети Флойда, содержит 19 компараторов.

Заметим, что программная реализация медианной фильтрации для окна 3×3 в библиотеке OpenCV основана на сортирующей сети, также содержащей 19 компараторов, и позволяет реализовать поиск медианы за 19 операций сравнения-обмена.

2. Программная реализация компараторов в сортирующих сетях

Значительным преимуществом алгоритмов на основе сортирующих сетей является возможность их реализации без использования команд условного перехода. Само понятие сортирующей сети подразумевает независимость каждого компаратора от предыдущих компараторов. Поэтому при выполнении последовательности действий, задаваемых сортирующей сетью, получим требуемый результат независимо от того, в каком порядке исходные данные поступали в сеть. Таким образом, на основе сортирующих сетей можно получить полностью линейный код без ветвлений, выполняемый на современных процессорах максимально быстро.

Одним из методов уменьшения количества ветвлений в программе является их замена арифметическими действиями.

Компаратор сортирующей сети можно реализовать программно, например

```
if ( a > b )
{
        v = a; a = b; b = v;
}
```

где a и b – входные значения; v – временная переменная.

В реализации компаратора можно обойтись без использования дополнительной временной переменной [16]:

```
if ( a > b )
{
        a = b – a; a = b – a; a = a + b;
}
```

Для обмена элементов без использования вспомогательной переменной также можно применять оператор XOR [16].

Между тем подобные программные реализации компаратора используют условный оператор, что значительно уменьшает быстродействие программы по сравнению с реализацией компаратора без применения условных операторов. Исключить ветвления из реализации компаратора можно при помощи ассемблера [11–15].

Еще одним вариантом реализации компаратора является применение выборки из таблицы, подобный подход использован в реализации медианного фильтра с ядром 3×3 в широко

применяемой на данный момент библиотеке компьютерного зрения OpenCV [24]. Сформируем таблицу T, элементы которой определяются следующим образом:

$$T_i = 0,\ i = \overline{0,255},\ T_i = i - 256,\ i = \overline{256,256 + 255 \times 2}.$$

Делая выборку по индексу $i = a - b + 256$ из таблицы T, получим

$$R = T[a - b + 256].$$

Таким образом, R будет равно нулю в случае, если $(a - b) \le 0$, и равно $(a - b)$, если $a > b$. Тогда, выполнив следующие арифметические действия: $a = a - R$, $b = b + R$, получим обмен значений a и b только в случае, если $a > b$.

Наиболее эффективным способом программной реализации сортирующей сети без применения условных операторов является использование операций MMX (Multimedia Extensions) и арифметики с насыщением [14, 25]. Применение MMX не только позволяет реализовывать компараторы без использования условных операторов, но и выполнять одновременно сразу несколько операций сравнения-обмена, что повышает скорость вычисления медианы в несколько раз. Однако необходимо заметить, что эффективное применение MMX для медианной фильтрации требует специально разработанного алгоритма (результаты по быстродействию алгоритмов, представленные в таблице, получены без использования MMX).

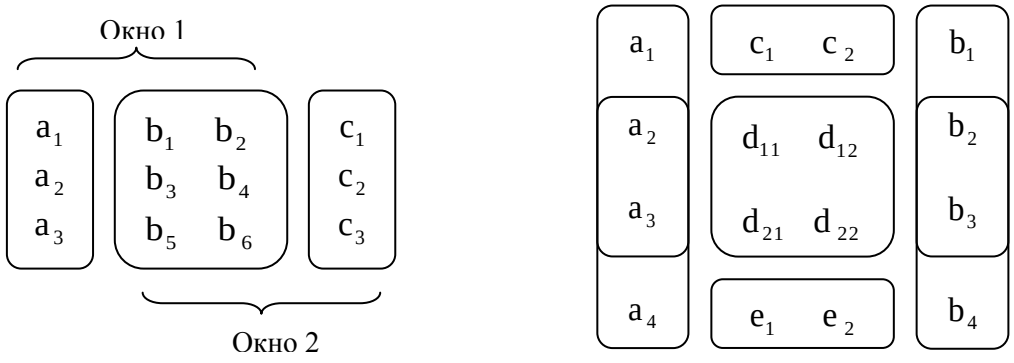
Таким образом, сортирующие сети допускают программную реализацию на персональном компьютере без применения ветвлений. Подобный линейный код значительно превосходит по быстродействию код с условными операторами.

Зачастую многие алгоритмы медианной фильтрации (слияния упорядоченных столбцов, неполной быстрой сортировки), теоретически требующие меньшего количества операций, но не допускающие простую реализацию без использования условных операторов, проигрывают по практическому быстродействию алгоритмам, основанным на сортирующих сетях и допускающих реализацию без ветвлений (см. таблицу). Например, алгоритмы медианной фильтрации слиянием упорядоченных столбцов и неполной сортировки прямым выбором выполняются приблизительно за одно и то же время, хотя первый требует 9,5 операции сравнения, а второй – 15 операций.

3. Сортирующие сети для пересекающихся окон фильтра

Неполные сортирующие сети поиска медианы на основе сортирующей сети Флойда и нечетно-четной сортировки Бэтчера позволяют найти медиану произвольных девяти элементов при минимальном количестве компараторов [3, 16]. При медианной фильтрации изображений можно учесть тот факт, что соседние окна медианного фильтра пересекаются. Например, для окна фильтра размером 3×3 соседние два окна фильтра имеют шесть общих элементов. Таким образом, для этого случая пересечения соседних окон фильтра можно построить более эффективные сортирующие сети, учитывающие тот факт, что некоторые операции сравнения общих элементов двух окон фильтра нет необходимости выполнять дважды для разных окон [6, 7, 17, 18].

Если рассматривать два соседних окна медианного фильтра (рис. 5, *а*), можно получить алгоритм, требующий 16,5 операции сравнения на поиск одной медианы [7, 19]. Если рассматривать четыре пересекающихся окна фильтра (рис. 5, *б*), количество операций сравнения можно уменьшить до 15 [7]. Также существует алгоритм на основе сортирующих сетей для поиска медианы окна фильтра, разработанный К.И. Кучеренко и Е.Ф. Очиным [17], который вычисляет медианы последовательно с учетом пересечения соседних окон фильтра (экономятся операции, необходимые на сортировку первых двух столбцов окна фильтра, выполненные для предыдущего окна). Алгоритм при выполнении медианной фильтрации изображения требует для каждого окна 3×3 около 13 операций сравнения-обмена.



а) *б)*

Рис. 5. Пересекающиеся части соседних окон медианного фильтра:
а) два соседних окна; *б)* четыре соседних окна

Алгоритм медианной фильтрации на основе неполной сортировки прямым выбором [7], легко реализуемый без применения условных операторов, также может быть описан в терминах сортирующих сетей. При этом используются сортирующие сети с одним отличием – способностью на некоторых этапах сохранять данные и применять их уже в других сетях.

Однако при рассмотрении двух пересекающихся соседних окон фильтра можно предложить алгоритм на основе сортирующих сетей, более эффективный по сравнению с описанными алгоритмами и требующий 12 операций сравнения-обмена для поиска медианы одного окна фильтра.

3.1. Алгоритм поиска медиан для двух соседних окон фильтра

В работе [7] поиск медиан двух соседних окон фильтрации был дан в терминах сортировки при помощи прямого выбора. Сначала сортировалась общая часть окон – элементы $b_1, b_2, b_3, b_4, b_5, b_6$ (рис. 5, *а*), затем части окон, не вошедших в эту общую часть, элементы a_1, a_2, a_3 и c_1, c_2, c_3. При этом сортировка велась до тех пор, пока не была найдена медиана соответствующего окна. Теперь будем описывать процесс поиска медианы при помощи сортирующих сетей.

Вначале построим эффективную сортирующую сеть для сортировки шести общих элементов соседних окон, потом сети для поиска медианы первого и второго окна на основе отсортированной общей части и оставшихся элементов окон.

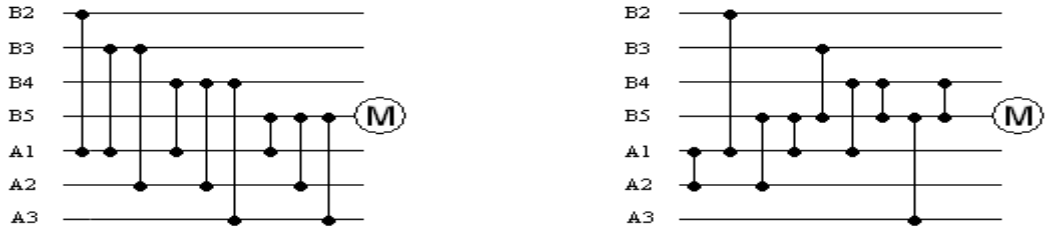
Пусть имеется два пересекающихся окна медианного фильтра (рис. 5, *а*). Общая их часть содержит элементы $b_1, b_2, b_3, b_4, b_5, b_6$, первый столбец первого окна фильтра – элементы a_1, a_2, a_3, последний столбец второго окна фильтра – элементы c_1, c_2, c_3. Рассмотрим сначала процедуру сортировки шести общих элементов окон фильтра $b_1, b_2, b_3, b_4, b_5, b_6$. Для сортировки этих элементов при помощи алгоритма прямого выбора требуется $5+4+3+2+1=15$ операций. Однако можно построить сортирующие сети, позволяющие сортировать 6 элементов за 12 операций сравнения-обмена (рис. 6).

а) *б)*

Рис. 6. Эффективные сети сортировки шести элементов: *а)* сеть на основе слияния отсортированной четверки и двойки элементов; *б)* сеть на основе слияния отсортированных троек элементов

Рассмотрим четыре первых элемента: b_1, b_2, b_3, b_4. Их сортировку при помощи сортирующей сети можно выполнить за пять операций. Слияние отсортированной четверки и отсортированной пары элементов b_5, b_6 можно осуществить за шесть операций. Основываясь на сортировке четырех элементов и слиянии их с парой оставшихся (одна операция на их упорядочивание) потребуется $5+1+6=12$ операций сравнения, что на три операции меньше, чем необходимо при сортировке прямым выбором (рис. 6, *а*). Аналогичного количества операций можно добиться, построив сортирующую сеть на основе сортировки троек и последующего их слияния (рис. 6, *б*).

Обозначим отсортированные шесть элементов $B_1, B_2, B_3, B_4, B_5, B_6$. Заметим, что из них медианой не может быть первый и последние элементы – B_1 и B_6, поэтому отбросим их и будем рассматривать отсортированную четверку элементов B_2, B_3, B_4, B_5. Если поиск двух медиан на основе отсортированной четверки и оставшихся элементов окон соседних фильтров выполнить при помощи способа, используемого в алгоритме поиска медианы при помощи прямого выбора [7], то на это потребуется еще 18 операций (9 + 9). Сеть, реализующая такой поиск медианы, изображена на рис. 7, *а*. Можно построить и другие сети, с помощью которых также можно получить аналогичный результат (рис. 7, *б*). Получим всего $12+18=30$ операций на поиск двух медиан. Следовательно, на поиск одной необходимо 15 операций.

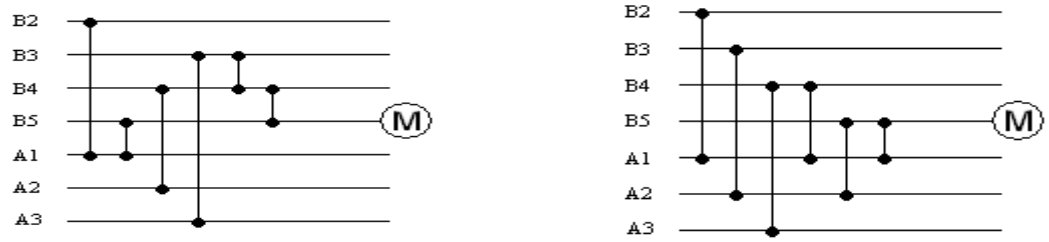


а) *б)*

Рис. 7. Сети поиска медианы на основе отсортированной четверки элементов и оставшихся трех элементов окна

3.2. Алгоритм поиска медиан для двух соседних окон фильтра на основе неполной сортирующей сети с памятью

Заметим, что если производить сортировку общей части двух окон на основе сортировки столбцов-троек (рис. 6, *б*), то можно отсортированные тройки использовать в дальнейшем для поиска медианы. Таким образом, можно на каждом шаге для следующих двух окон фильтра сортировать два новых столбца из трех элементов, а два других уже будут отсортированы на предыдущем шаге алгоритма. Сортировка общей части двух окон с учетом сортировки двух столбцов потребует 12 операций: 3+3 для сортировки столбцов и 6 для слияния двух отсортированных столбцов. Тогда для поиска медиан двух соседних окон на основе отсортированной четверки элементов B_2, B_3, B_4, B_5 и оставшихся трех отсортированных элементов каждого окна можно построить более эффективные сети поиска медиан (рис. 8).



а) *б)*

Рис. 8. Более эффективные сети поиска медианы на основе отсортированной четверки элементов и оставшихся трех отсортированных элементов окна

Рассмотрим сеть, изображенную на рис. 8, *а*. Пусть элементы B_2, B_3, B_4, B_5 отсортированы, как и элементы A_1, A_2, A_3. Так как элемент B_1 не рассматривается при поиске медианы, нам необходимо из перечисленных выше элементов найти элемент, который будет находиться на четвертом месте в общей отсортированной последовательности из элементов $B_2, B_3, B_4, B_5, A_1, A_2, A_3$. Очевидно, что минимум из элементов B_2 и A_1 не может быть медианой, так как он является минимальным элементом общей последовательности. Таким образом, первой операцией сравнения-обмена отсеивается один из элементов и дальше не рассматривается. Затем следующими тремя операциями сравнения-обмена отбрасываются еще три элемента, так как каждый найденный при этих сравнениях максимальный элемент является минимум пятым в результирующей отсортированной последовательности. В результате остаются три элемента-кандидата на роль четвертого элемента, так как найден один элемент, стоящий до этих трех элементов, и три, стоящие после них. Тогда четвертым элементом в результирующем списке будет максимальный из этих трех оставшихся элементов. Для поиска максимума из трех элементов необходимы еще две операции сравнения-обмена.

На рис. 8, *б* изображена сортирующая сеть, также выполняющая поиск четвертого элемента на основе отсортированной четверки и отсортированной тройки элементов.

В итоге имеем 12 операций, необходимых для сортировки двух очередных столбцов и слияния с получением отсортированной четверки элементов, и по 6 операций для слияния отсортированной четверки с левым и правым отсортированными столбцами. Всего 12 + 6 + 6 = 24 операции на поиск двух медиан; следовательно, на поиск одной медианы требуется 12 операций сравнения-обмена.

Получим *алгоритм медианной фильтрации при помощи неполной сортирующей сети с памятью* для трех первых строк фильтруемого изображения (остальные строки обрабатываются аналогично):

1. Сортируем два первых столбца по три элемента, отсортированные значения сохраняем в переменные S_1, S_2.

2. Сортируем два очередных столбца по три элемента, результат сохраняем в переменные S_3, S_4, а также во вспомогательные переменные V_3, V_4.

3. Осуществляем слияние двух средних столбцов из четырех – S_2 и S_3 – при помощи сортирующей сети, изображенной на рис. 6, *б*, и сохраняем четыре средних элемента результирующего массива b_2, b_3, b_4, b_5 во вспомогательные переменные v_2, v_3, v_4, v_5.

4. Осуществляем при помощи сортирующей сети, изображенной на рис. 8, *а* или рис. 8, *б*, поиск медианы для левого окна из двух пересекающихся окон фильтра, используя элементы b_2, b_3, b_4, b_5 и столбец S_1.

5. Осуществляем при помощи сортирующей сети, изображенной на рис. 8, *а* или рис. 8, *б*, поиск медианы для правого окна из двух пересекающихся окон фильтра, используя элементы v_2, v_3, v_4, v_5 и столбец S_4.

6. Копируем в столбцы S_1, S_2 значения, сохраненные в переменных V_3, V_4, для использования на следующих итерациях алгоритма.

7. Рассматриваем два следующих столбца и повторяем шаги 2–6 до тех пор, пока не будут найдены все медианы.

Описанный алгоритм на основе неполных сортирующих сетей схож с алгоритмом слияния упорядоченных столбцов [18, 19], однако в отличие от него допускает простую полностью линейную реализацию без условных операторов. Алгоритм слияния упорядоченных столбцов позволяет получить медиану окна фильтра за 9,5 операции сравнения, что на 2,5 операции меньше, чем необходимо описанному выше алгоритму. Преимущество алгоритма слияния упорядоченных столбцов по количеству операций достигается за счет эффективной процедуры слияния отсортированных последовательностей. Алгоритм слияния на каждом своем шаге учитывает результат предыдущего шага, что привносит в алгоритм элемент логики и поэтому по-

добное слияние весьма сложно реализовать при помощи сортирующей сети. Существующие сети слияния упорядоченных последовательностей содержат заведомо больше операций сравнения, нежели процедура слияния при помощи условных операторов. Алгоритм слияния упорядоченных столбцов допускает реализацию без ветвлений (не через сортирующие сети), но реализация эта является более сложной и более трудоемкой в вычислительном плане, вследствие чего алгоритм проигрывает по быстродействию предложенному выше (см. таблицу). Заметим, что алгоритм Кучеренко – Очина, изначально предназначавшийся для аппаратной реализации, был реализован автором данной статьи без использования условных операторов при помощи описанных ранее методов (что не предусматривалось авторами алгоритма). Это значительно повысило его быстродействие и позволило приблизиться по времени выполнения к алгоритму медианной фильтрации на основе неполных сортирующих сетей с памятью.

Производительность и характеристики предложенных алгоритмов медианной фильтрации с окном 3×3 в сравнении с другими алгоритмами отображены в таблице. Тестирование алгоритмов производилось на полутоновом изображении размером 1024×768 при помощи персонального компьютера с процессором Intel Pentium 4 с тактовой частотой 3 ГГц.

Таблица

Быстродействие различных алгоритмов медианной фильтрации для полутонового изображения размера 1024×768 пикселей ($|d|$ – среднее отклонение медиан соседних окон фильтра в изображении)

Алгоритм поиска медианы	Количество операций сравнения	Быстродействие, мс	
		Реализация с ветвлениями	Реализация без ветвлений
Алгоритм слияния упорядоченных столбцов	9,5	72	23
Неполная сортировка прямым выбором	15	66	18
Алгоритм медианной фильтрации Хуанга	$7{,}5+\|d\|$	71	50
Алгоритм Паеса	20	90	29
Неполная сортирующая сеть на основе сети Флойда	19	80	28
Неполная сортирующая сеть на основе сети Бэтчера	19	90	28
OpenCV (3x3)	19	–	72
Модификация алгоритма Кучеренко – Очина	13	58,6	15,7
Неполная сортирующая сеть с памятью	12	54,5	15,2

Заметим, что предложенный алгоритм медианной фильтрации при помощи неполных сортирующих сетей можно усовершенствовать, рассматривая пересечение не двух соседних окон фильтра, а четырех (двух по горизонтали и двух по вертикали). Тогда вместо сортировки столбцов по три элемента можно использовать сортировку столбцов по четыре элемента, что можно сделать за пять, а не за шесть операций. Таким образом, в среднем экономится 0,5 операции сравнения на одну медиану. В этом случае количество операций, необходимых для поиска медианы каждого окна фильтра, сократится до 11,5, а скорость работы медианного фильтра возрастет.

Заключение

Построены неполные сортирующие сети на основе сортирующей сети Флойда и сортирующей сети Бэтчера для поиска медианы из девяти элементов 3×3 окна фильтра с минимально возможным количеством компараторов – 19. Предложен алгоритм медианной фильтрации изображений на основе неполных сортирующих сетей, позволяющий находить медиану окна фильтра за 12 операций сравнения-обмена, что превосходит известные алгоритмы поиска медианы на основе сортирующих сетей и операций сравнения-обмена. Алгоритм может быть программно реализован без использования условных операторов, что значительно повышает его быстродействие и позволяет применять для обработки изображений в режиме реального времени.

Список литературы

1. Шапиро, Л. Компьютерное зрение / Л. Шапиро, Дж. Стокман. – М. : БИНОМ. Лаборатория знаний, 2006. – 752 с.

2. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М. : Техносфера, 2005. – 1072 с.

3. Vasicek, Z. Novel Hardware Implementation of Adaptive Median Filters / Z. Vasicek, L. Sekanina // IEEE Workshop «Design and Diagnostics of Electronic Circuits and Systems» (DDECS'08). – Bratislava, 2008. – P. 1–6.

4. Perreault, S. Median Filtering in Constant Time / S. Perreault, P. Hébert // IEEE Transactions on Image Proc. – 2007. – Vol. 16, № 9. – P. 2389–2394.

5. Weiss, B. Fast median and bilateral filtering / B. Weiss // ACM Transactions on Graphics. – 2006. – Vol. 25, № 3. – P. 519–526.

6. Быстрые алгоритмы в цифровой обработке изображений / Т.С. Хуанг [и др.] ; под общ. ред. Т.С. Хуанга. – М. : Радио и связь, 1984. – 220 с.

7. Кравчонок, А.И. Алгоритмы медианной фильтрации с окном 3×3 на основе неполной сортировки прямым выбором / А.И. Кравчонок // Информатика. – 2008. – № 1(17). – С. 38–46.

8. Кучеренко, К.И. Двумерные медианные фильтры для обработки изображений / К.И. Кучеренко, Е.Ф. Очин // Зарубежная радиоэлектроника. – 1986. – № 6. – С. 50–61.

9. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт. – СПб. : Невский диалект, 2005. – 360 с.

10. Кнут, Д. Искусство программирования. Т. 3. Сортировка и поиск / Д. Кнут. – М. : Издательский дом «Вильямс», 2005. – 824 с.

11. Касперски, К. Техника оптимизации программ. Эффективное использование памяти / К. Касперски. – СПб. : БХВ – Петербург, 2003. – 464 с.

12. Магда, Ю.С. Аппаратное обеспечение и эффективное программирование / Ю.С. Магда. – СПб. : Питер, 2007. – 352 с.

13. Магда, Ю.С. Ассемблер для процессоров Intel Pentium / Ю.С. Магда. – СПб. : Питер, 2006. – 410 с.

14. Магда, Ю.С. Использование ассемблера для оптимизации программ на C++ / Ю.С. Магда. – СПб. : БХВ-Петербург, 2004. – 496 с.

15. Юров, В.И. Assembler. Практикум / В.И. Юров. – 2-е изд. – СПб. : Питер, 2006. – 399 с.

16. Paeth, A. Median Finding of a 3×3 Grid / A. Paeth, W. Alan // Graphics Gems I / S. Andrew (ed.). – Academic Press, 1990. – P. 171–175.

17. Кучеренко, К.И. Сортирующие сети двумерной медианной фильтрации полутоновых изображений / К.И. Кучеренко, Е.Ф. Очин // Радиотехника. – 1987. – № 7. – С. 36–38.

18. Kravchonok, A. An Algorithm for Median Filtering on the Basis of Merging of Ordered Columns / A. Kravchonok, B. Zalesky, P. Lukashevich // Pattern Recognition and Image Analysis. – 2007. – Vol. 17, № 3. – P. 402–407.

19. Kopp, M. Efficient 3×3 Median Filter Computations / M. Kopp // Machine Graphics & Vision. – 1995. – Vol. 4, № 1/2. – P. 79–82.

20. Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. – 2-е изд. – М. : Издательский дом «Вильямс», 2007. – 1296 с.

21. Cormen, T. Introduction to algorithms / T. Cormen, C. Leiserson, R. Rivest. – MIT Press, Cambridge, MA, 2001. – 984 p.

22. Седжвик, Р. Фундаментальные алгоритмы на C. Анализ/Структуры/Сортировка/Поиск : пер. с англ. / Р. Седжвик. – СПб. : ООО «Диа Софт ЮП», 2003. – 672 с.

23. Миллер, Р. Последовательные и параллельные алгоритмы / Р. Миллер, Л. Боксер. – М. : БИНОМ. Лаборатория знаний, 2006. – 406 с.

24. Open Computer Vision Library / Sourceforge.net. Open Source Software [Electronic resource]. – 1999. – Mode of access : http://sourceforge.net/projects/opencvlibrary/. – Date of access : 28.01.2009.

25. Зубков, С.В. Assembler для DOS, Windows и UNIX / С.В. Зубков. – 3-е изд. – М. : LVR Пресс; СПб. : Питер, 2006. – 608 с.

Поступила 17.12.08

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: alpha_storm@mail.ru

A.I. Kravchonok

ALGORITHMS FOR MEDIAN FILTERING WITH 3×3 WINDOW ON THE BASIS OF INCOMPLETE SORTING NETWORKS

New algorithms of median filtering with 3×3 window based on the use of incomplete sorting networks that implement an efficient search of median elements are proposed. The use of algorithms increases the speed of median filtering so that it becoming possible to solve various problems of image processing in real time. A comparative analysis of different algorithms of median filtering considering the number of operations necessary for their performance and speed is performed.